THE FAMILY OF STABLE MODELS

MELVIN FITTING

▷ The family of all stable models for a logic program has a surprisingly simple overall structure, once two naturally occurring orderings are made explicit. In a so-called knowledge ordering based on degree of definedness, every logic program $\mathscr{P}$ has a smallest stable model $s^k_{\mathscr{P}}$—it is the well-founded model. There is also a dual largest stable model $S^k_{\mathscr{P}}$, which has not been considered before. There is another ordering based on degree of truth. Taking the meet and the join, in the truth ordering, of the two extreme stable models $s^k_{\mathscr{P}}$ and $S^k_{\mathscr{P}}$ just mentioned yields the alternating fixed points of Van Gelder, denoted $s^t_{\mathscr{P}}$ and $S^t_{\mathscr{P}}$ here. From $s^t_{\mathscr{P}}$ and $S^t_{\mathscr{P}}$ in turn, $s^k_{\mathscr{P}}$ and $S^k_{\mathscr{P}}$ can be produced again, using the meet and joint of the knowledge ordering. All stable models are bounded by these four valuations. Further, the methods of proof apply not just to logic programs considered classically, but to logic programs over any bilattice meeting certain conditions, and thus apply in a vast range of settings. The methods of proof are largely algebraic. ◁

1. INTRODUCTION

Stable model semantics [17, 6] and well-founded model semantics [30, 31] have provided valuable insights on the meaning of logic programs. While much is known about these semantics, it has not been made clear just how elegant a structure the family of stable models has. The delineation of this structure is the primary aim of this paper. We do not attempt to state our results rigorously at this point, but wait until Section 9, after appropriate terminology and concepts have been introduced. We can give the general flavor, however.

As originally formulated, stable model semantics was classical, two-valued. Under these circumstances not every program has a stable model. Three-valued, or

Address correspondence to Melvin Fitting, Departments of Computer Science, Philosophy, and Mathematics, Graduate Center (CUNY), 33 West 42nd Street, New York, NY 10036. Email: mlflc@cunyvm.cuny.edu.

Received June 1992; accepted May 1993.

655 Avenue of the Americas, New York, NY 10010 0743-1066/93/$6.00

partial, model semantics has had an extensive development for logic programs generally [7, 20, 21]. Soon Przymusinski extended the notion of stable model to allow three-valued, or partial, stable models, [22, 23], and showed every program has at least one partial stable model, and the well-founded model is the smallest among them, in a natural ordering [24]. Once one has made the transition from classical to partial models allowing incomplete information, it is a small step also to allow models admitting inconsistent information. Doing so provides a natural framework for the semantic understanding of logic programs that are distributed over several sites, with possibly conflicting information coming from different places [10]. We make the extension to such a setting in this paper, as part of a much more general investigation.

One can formally think of allowing incompleteness or inconsistency of information in logic programs in the following way. Take a valuation to be a mapping from ground atoms to *sets* of truth values. The values {*true*} and {*false*} correspond to the usual notions of truth and falsity, while $\emptyset$ and {*false, true*} correspond to no information and inconsistent information, respectively. These are the four truth values of a logic due to Belnap [3], one that contains the Kleene strong three-valued logic as a sublogic. Now the space of valuations has two natural orderings. One is on the "degree of truth." In this ordering an increase means that one or more ground atoms loses an assignment of *false* or gains an assignment of *true*. The other ordering is on the "degree of knowledge." In this ordering an increase means at least one ground atom gains an assignment of *true* or *false* that it did not have, without losing any value it did have. As we will see, these two orderings are intimately connected. Now, we will show that the family of stable models for a program is bounded from above and below, in the knowledge ordering, by a biggest and a smallest stable model. The smallest is the well-founded model; this is Przymusinski's result. The existence of a largest stable model is new and could not have been formulated without moving to the setting of Belnap's logic. Also, the family of stable models is bounded from above and below in the truth ordering by two valuations that are not necessarily stable models themselves, but which are extremal in a certain sense. This, essentially, is the "alternating fixpoint construction" of Van Gelder [29]. Finally, we will show that the smallest and biggest stable models in the knowledge ordering and the extremal valuations in the truth ordering are closely connected, and that they can be "calculated" from each other. This is new.

The results just sketched actually hold in a much broader setting than was indicated. Establishing them in this broader setting takes no more work than it does in the classical context. In addition, working in the extended context tends to strip away some of the unnecessary detail and makes the underlying simplicity of the proofs stand out. We briefly outline the setting we have in mind.

The four-valued logic of Belnap is the simplest example of a nontrivial *bilattice*, a notion introduced by Matt Ginsberg in [19]. We have made use of bilattices in previous work on the semantics of logic programming [11–13], and once again we find them the proper tool. Motivation will be supplied in more detail in Section 4. For now, we merely remark that bilattices arise naturally in two ways, which we sketch briefly.

First, instead of the all-or-nothing approach of classical logic, we might introduce a more general notion of evidence, such as a confidence factor. Then, further, we might separate the roles of positive and negative evidence. This leads directly to

bilattices and is the approach presented in [11] and [12]. From this point of view, classical logic programming is based on there being only two kinds of evidence: totally convincing and totally unconvincing.

Alternatively, we might think of the underlying logic as being more general than two-valued. For instance, if we have a program divided among two sites, then an answer to a ground query should contain information on which sites answer yes, and so there are four natural truth values: neither, one, the other, both. We might even allow dependencies between sites. This gives rise to an algebraic structure of truth values [25, 15]. If, further, we admit the possibility of incomplete or inconsistent information about the behavior of these sites, we once again get the structure of a bilattice [14]. A summary of this and the previous approach will be presented in Section 4.

The work in this paper extends results previously presented in [13]. At the time the earlier paper was written, the real simplicity of the structure of stable models had not become clear; hence the need for this fuller presentation. We remark again that moving from the classical setting to that of bilattices adds no complications. Indeed, we believe it makes the underlying simplicity of the results stand out by removing unessential details.

2. BELNAP'S LOGIC, $\mathscr{FOUR}$

In [3], Belnap introduced a logic intended to deal in a useful way with inconsistent or incomplete information. It is the simplest example of a nontrivial bilattice and it illustrates many of the basic ideas concerning them. In the next section we present the general notion of bilattice; in the following one we show there are lots of them and they arise in natural ways. In fact, Belnap's logic is a good enough representative of the whole family that, if you wish, on a first reading of this paper you could skip from the end of this section directly to Section 5, just assuming that every bilattice discussed is sufficiently like the Belnap one.

We can think of Belnap's values as *sets* of ordinary truth values. Suppose we write, simply, *true* for $\{true\}$ and *false* for $\{false\}$. Also we write $\bot$ for $\emptyset$ and think of it as indicating a lack of information; we write $\top$ for $\{false, true\}$ and think of it as indicating inconsistency.

The truth values of Belnap's logic have two natural orderings. One is the subset relation. For instance, $\bot = \emptyset \subset \{false\} = false$. If $A \subseteq B$, in a natural sense B represents more information than A, so this is referred to as a *knowledge* ordering and is denoted by $\leq_k$. Thus $\bot \leq_k false$. The other ordering is on the *degree of truth* and is denoted $\leq_t$. Here $A \leq_t B$ if "B is at least as true as A is, and A is at least as false as B is." More precisely, $A \leq_t B$ if $A \cap \{true\} \subseteq B \cap \{true\}$ and $B \cap \{false\} \subseteq A \cap \{false\}$. Under this ordering, $false \leq_t \bot$. The two orderings are represented in the double Hasse diagram shown in Figure 1.

Both $\leq_k$ and $\leq_t$ give $\mathscr{FOUR}$ a lattice structure. Meet and join under $\leq_t$ are denoted $\wedge$ and $\vee$; they are natural generalizations of the usual conjunction and disjunction notions. Meet and join under $\leq_k$ are denoted $\otimes$ and $\oplus$; $x \otimes y$ is the most information x and y can agree on—we call $\otimes$ a *consensus* operator. Likewise $x \oplus y$ simply combines the knowledge represented by x with that represented by y, without checking for consistency—we call $\oplus$ a *gullability* operator. The four operations $\wedge$, $\vee$, $\otimes$, and $\oplus$ are intimately connected: all 12 distributive laws hold.

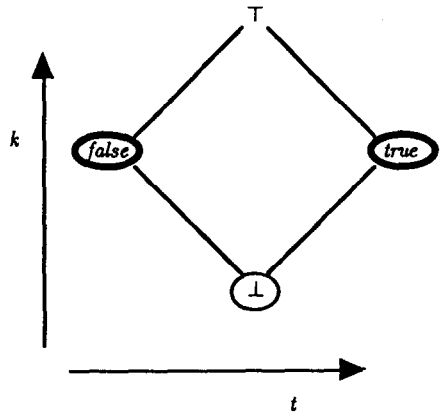

FIGURE 1. The logic $\mathscr{FOUR}$.

There is a natural notion of negation: flip the diagram from left to right, switching *false* and *true*, leaving $\bot$ and $\top$ alone. This is denoted $\neg$. There is also a natural symmetry vertically, switching $\bot$ and $\top$, leaving *false* and *true* alone. This is denoted $-$ and is called *conflation*. Conflation and negation commute: $-\neg x = \neg -x$.

The role of conflation is less immediate than that of negation. Notice that the truth values left unchanged under conflation are *false* and *true*, the classical ones. These are marked with dark circles in Figure 1. The operations $\neg$, $\wedge$, and $\vee$, restricted to the values *false* and *true*, reduce to those of classical logic. In addition, the values of $\mathscr{FOUR}$ that contain no more information than their conflation ($x \leq_k -x$) are *false*, *true*, and $\bot$, circled in Figure 1. It turns out that the operations $\wedge$, $\vee$, and $\neg$, restricted to *false*, *true*, and $\bot$, are exactly those of Kleene's strong three-valued logic. This way of distinguishing natural sublogics of $\mathscr{FOUR}$, using conflation, is something that carries over to bilattices generally.

3. BILATTICE DEFINITIONS

Now we turn to the general notion of bilattice—keep $\mathscr{FOUR}$ in mind as a representative example. A bilattice is a lattice-like structure with two inter-related orderings. Loosely, one of the orderings represents degree of truth, the other represents degree of knowledge, as with $\mathscr{FOUR}$. We sketch ways bilattices arise in the next section, after the formal definitions have been presented. One can make assumptions of various strengths concerning relationships between the orderings—in this way one obtains different kinds of bilattices. We will need rather strong assumptions for this paper.

Definition 3.1. A *pre-bilattice* is a structure $\langle \mathscr{B}, \leq_t, \leq_k \rangle$, where $\mathscr{B}$ is a nonempty set and $\leq_t$ and $\leq_k$ are each partial orderings giving $\mathscr{B}$ the structure of a lattice with a top and a bottom.

REMARK. As originally defined, in [19], both $\leq_t$ and $\leq_k$ were required to yield *complete* lattices. In various subsequent papers completeness has sometimes been assumed, sometimes not. Generally it has been assumed if an application involved quantifiers, but it was not assumed if algebraic properties of bilattices were being

studied. We feel it is time to end this confusion. In this paper we do not assume completeness without saying so. Thus a pre-bilattice will specifically be called *complete* if its partial orderings give the structure of complete lattices. Similarly for interlaced and distributive bilattices below.

Definition 3.2. In a pre-bilattice $\langle \mathscr{B}, \leq_t, \leq_k \rangle$, meet and join under $\leq_t$ are denoted $\wedge$ and $\vee$, and meet the join under $\leq_k$ are denoted $\otimes$ and $\oplus$. Top and bottom under $\leq_t$ are denoted *true* and *false*, and top and bottom under $\leq_k$ are denoted $\top$ and $\bot$. If the pre-bilattice is complete, infinitary meet and joint under $\leq_t$ are denoted $\bigwedge$ and $\bigvee$, and infinitary meet and join under $\leq_k$ are denoted $\prod$ and $\sum$.

The operations $\wedge$ and $\vee$ should be thought of as generalizations of conjunction and disjunction in the classical space {*false*, *true*}. The operations $\otimes$ (consensus) and $\oplus$ (gullability) are not meaningful in the classical setting. One needs at least the structure of $\mathscr{FOUR}$ for them to arise. Now we define the various basic notions we will be using here.

Definition 3.3. A *distributive bilattice* is a pre-bilattice $\langle \mathscr{B}, \leq_t, \leq_k \rangle$ in which all 12 distributive laws connecting $\wedge$, $\vee$, $\otimes$, and $\oplus$ hold. An *infinitely distributive bilattice* is a complete pre-bilattice in which all infinitary, as well as all finitary, distributive laws hold.

An example of a distributive law is $x \otimes (y \vee z) = (x \otimes y) \vee (x \otimes z)$. An example of an infinitary distributive law is $x \otimes \bigwedge\{y_i | i \in S\} = \bigwedge\{x \otimes y_i | i \in S\}$. There is a weaker notion than distributivity that also has played a role in the semantics of logic programming [11, 12].

Definition 3.4. The pre-bilattice $\langle \mathscr{B}, \leq_t, \leq_k \rangle$ satisfies the *interlacing conditions* if each of the lattice operations $\wedge, \vee, \otimes, \oplus$ is monotone with respect to both orderings. If the pre-bilattice is complete, it satisfies the *infinitary interlacing conditions* if each of the infinitary meet and join operations is monotone with respect to both orderings.

An example of an interlacing condition is: $x_1 \leq_t y_1$ and $x_2 \leq_t y_2$ implies $x_1 \otimes x_2 \leq_t y_1 \otimes y_2$. An example of an infinitary interlacing condition is: $x_i \leq_t y_i$ for each $i \in S$ implies $\prod\{x_i | i \in S\} \leq_t \prod\{y_i | i \in S\}$. A distributive bilattice meets the interlacing conditions as well. This is easily verified. However, an infinitely distributive bilattice need not meet the infinitary interlacing conditions. We will explicitly assume such conditions, when needed.

As defined in [19], bilattices were required to have a negation operation, and the connection between the two orderings was a very simple one, expressed via negation. We find this too weak for our purposes, though negation still plays an important role here, along with the "dual" operation of conflation. In the following, *bilattice* refers to any of the above: distributive bilattice, complete distributive bilattice, interlaced bilattice, or complete interlaced bilattice.

Definition 3.5. A bilattice has a *negation* if there is a mapping $\neg$ that reverses the $\leq_t$ ordering, leaves unchanged the $\leq_k$ ordering, and $\neg \neg x = x$. Similarly a bilattice has a *conflation* if there is a mapping $-$ that reverses the $\leq_k$ ordering,

leaves unchanged the $\leq_t$ ordering, and $--x = x$. If a bilattice has both, we say they *commute* if $-\neg x = \neg -x$ for all x.

Bilattices can have negations without conflations, conflations without negations, neither, or both. Examples are not hard to come by.

4. BILATTICE CONSTRUCTIONS

Bilattices come up in natural ways—indeed the simplest nontrivial bilattice, $\mathcal{FOUR}$, was already introduced in [3]. Two general, but different, constructions have appeared in the literature. We sketch them briefly here in order to give a feeling for bilattices and their possible applications.

The first construction comes from [19] and is completely general for distributive bilattices—that is, a representation theorem can be proved. Here we follow the presentation of [11].

Suppose we have two lattices, $\langle L_1, \leq_1 \rangle$ and $\langle L_2, \leq_2 \rangle$. Think of L_1 as the lattice of values we use when we measure the degree of belief, evidence, confidence, etc. that we have in a sentence. Likewise, think of L_2 as what we use when we measure the degree of doubt, counter-evidence, lack of confidence, etc. that we have against a sentence. For instance, L_1 and L_2 could both be just {*false*, *true*}, with *false* $\leq$ *true*, where we have all-or-nothing judgments. Or each could be the unit interval, with the standard ordering. Or L_1 could be the unit interval and L_2 could be {*false*, *true*}, a choice that is appropriate for a theoretical science, where one has a degree of confidence in a sentence, but a single counter-experiment is enough to falsify. Or again, L_1 and L_2 could be sets of experts, ordered by inclusion (the {*false*, *true*} example can be thought of as arising from a single expert).

Now we define a structure $L_1 \odot L_2$ as follows. The structure is $\langle L_1 \times L_2, \leq_t, \leq_k \rangle$, where:

- $\langle x_1, x_2 \rangle \leq_t \langle y_1, y_2 \rangle$ if $x_1 \leq_1 y_1$ and $y_2 \leq_2 x_2$;
- $\langle x_1, x_2 \rangle \leq_k \langle y_1, y_2 \rangle$ if $x_1 \leq_1 y_1$ and $x_2 \leq_2 y_2$.

The idea is that knowledge goes up if evidence, both for and against, goes up; degree of truth goes up if evidence for goes up while evidence against goes down.

It is straightforward to verify that $L_1 \odot L_2$ will always be a bilattice meeting the interlacing conditions, and if L_1 and L_2 are complete, $L_1 \odot L_2$ will be a complete bilattice meeting the infinitary interlacing conditions as well. Further, if both L_1 and L_2 are distributive lattices, $L_1 \odot L_2$ will be a distributive bilattice, and if L_1 and L_2 are complete and infinitely distributive, $L_1 \odot L_2$ will be an infinitely distributive bilattice. This can be carried yet further. If $L_1 = L_2 = L$, then we are measuring belief and doubt the same way, and so a negation can be introduced into the bilattice $L \odot L$ in the obvious way: set $\neg \langle x, y \rangle = \langle y, x \rangle$. This is easily seen to meet the appropriate conditions for a negation operation. The intuition is straightforward: In passing from a member of $L \odot L$ to its negation we simply switch the roles of belief and doubt. Finally, suppose $L_1 = L_2 = L$, and further suppose that L has an order reversing involution (we denote the involute of x by $\bar{x}$). Such an operation is called a *de Morgan complement*. Then a conflation operation can be introduced into $L \odot L$ as well. Set $-\langle x, y \rangle = \langle \bar{y}, \bar{x} \rangle$. Once again it is easy to verify

that this meets the conditions for a conflation operation, and conflation and negation will commute.

The idea behind this notion of conflation is a little more complicated than that behind negation. Think, for example, of L as being the unit interval, with members measuring degree of belief or doubt. Set $\bar{x} = 1 - x$, which is a de Morgan complement. Then $-\langle x, y\rangle = \langle 1-y, 1-x\rangle$, and so in passing from a member of $L \odot L$ to its conflation, we replace the degree of belief by the degree to which we did not doubt, and we replace the degree of doubt by the degree to which we did not believe. Many other examples of this nature can be given.

A *de Morgan lattice* is a distributive lattice with a de Morgan complement. Thus we have the following: If L is a de Morgan lattice, $L \odot L$ is a distributive bilattice with a negation and a conflation that commute.

The simplest nontrivial de Morgan lattice is $L = \{\mathit{false}, \mathit{true}\}$, of course, with $\mathit{false} \leq \mathit{true}$. The corresponding bilattice, $L \odot L$, is just $\mathcal{FOUR}$ all over again, with $\bot = \langle \mathit{false}, \mathit{false}\rangle$, $\top = \langle \mathit{true}, \mathit{true}\rangle$, $\mathit{false} = \langle \mathit{false}, \mathit{true}\rangle$, and $\mathit{true} = \langle \mathit{true}, \mathit{false}\rangle$. The bilattice $\mathcal{FOUR}$ has a long history. It was introduced in [4], outfitted with two orderings in [3], and recognized as a bilattice in [19].

The method just sketched for constructing bilattices is general for the distributive case. That is, one can show representation theorems along the following lines. If $\mathcal{B}$ is a distributive bilattice, it is isomorphic to $L_1 \odot L_2$, where L_1 and L_2 are distributive lattices. If $\mathcal{B}$ has a negation, L_1 and L_2 can be taken to be the same lattice. If $\mathcal{B}$ has a negation and a conflation that commute, L_1 is a de Morgan lattice. Proofs of this can be found in [19], [11], and [16]. In fact, the result can be looked at as a variation on the polarities theorem of Dunn, which goes back to his dissertation of 1966 (see [5]).

A bilattice with conflation and negation that commute with each other contains some natural sublogics within it. Suppose $\mathcal{B}$ is such a bilattice. Call a member $x \in \mathcal{B}$ *consistent* if $x \leq_k -x$, and call x *exact* if $x = -x$. Just as with $\mathcal{FOUR}$, the exact members of a bilattice always form a generalization of the classical truth values. In particular, they are closed under $\wedge$, $\vee$, and $\neg$. In the bilattice $L \odot L$, where L is the unit interval, for instance, the exact members are those of the form $\langle x, y\rangle$, where $y = 1 - x$; thus they are the ones for which doubt exactly complements belief. The consistent members of a bilattice also are closed under $\wedge$, $\vee$, and $\neg$. In addition they are closed under $\otimes$, and under $\oplus$ when applied to a set that is directed under $\leq_k$. In [9] these facts were used to generalize Kripke's theory of truth to bilattices, and in [12] they played a role in developing a semantics for logic programming. In the bilattice based on the unit interval, mentioned above, the consistent values are those $\langle x, y\rangle$ for which $x + y \leq 1$, and thus represent situations where belief and doubt are not contradictory.

There is a second way in which bilattices arise naturally. It was sketched in [19] and developed in detail in [14]. The idea is to think of the consistent members as "approximations" to the exact values. This approach leads directly to Kleene's three-valued logic and indirectly to $\mathcal{FOUR}$, rather than the other way around, as above.

Suppose we have a lattice of truth values L. Think of these as the "real" values we are interested in. In classical logic the intended lattice is $\{\mathit{false}, \mathit{true}\}$. If we have two independent experts, say A and B, we have a more complicated lattice to work with, because the experts might disagree. In this case the natural lattice of truth values to work with is $\{\varnothing, \{A\}, \{B\}, \{A, B\}\}$, ordered by inclusion, where we think of

a set of experts as a representation that those experts said "yes" to a query. Other lattices of truth values arise if we allow probabilities as truth values, and so on.

In general we may be uncertain about the status of a sentence: what truth value in L to assign to it. Our ignorance may not be total, however; we may have some information to work with. In [14] approximations to a "real" truth value were represented by intervals containing that truth value.

Definition 4.1. Let $a, b \in L$, where L is a lattice, and assume $a \leq_L b$. The *interval* $[a, b]$ is $\{x \in L | a \leq_L x \leq_L b\}$. $\mathscr{I}(L)$ is the set of all intervals in L.

Now we take the space of intervals and give it a bilattice-like structure.

Definition 4.2. Let L be a lattice. $\mathscr{K}(L)$ is the structure $\langle \mathscr{I}(L), \leq_t, \leq_k \rangle$, where the following hold for $[a, b], [c, d] \in \mathscr{I}(L)$:

(1) $[a, b] \leq_t [c, d]$ if $a \leq_L c$ and $b \leq_L d$;
(2) $[a, b] \leq_k [c, d]$ if $[c, d] \subseteq [a, b]$.

The idea is, knowledge increases if the approximation interval narrows, and degree of truth increases if the interval shifts upward in the ordering of L.

This can be carried a little further. Suppose L also has an order-reversing involution, a de Morgan complement (again we denote the de Morgan complement of x by $\bar{x}$). Then a negation can be introduced in $\mathscr{K}(L)$ by

$$\neg[a, b] = [\bar{b}, \bar{a}].$$

As an example, suppose we start with the lattice of classical truth values, $L = \{false, true\}$. Then $\mathscr{K}(L)$ has three members, two one-point intervals, $[false, false]$ and $[true, true]$, and the entire lattice, $[false, true]$. The entire lattice represents a state of no information—denote it by $\perp$. Likewise $[false, false]$ can be identified with *false*, and $[true, true]$ with *true*. Then the ordering $\leq_t$ of $\mathscr{K}(L)$ yields operations $\wedge$ and $\vee$ that are those of Kleene's strong three-valued logic, and the negation operation on $\mathscr{K}(L)$ is likewise that of Kleene's logic. The $\leq_k$ ordering puts $\perp$ below both *false* and *true*, which are incomparable. It is the natural notion of approximation in this context.

Kleene's three-valued logic, it was noted above, is a sublogic of $\mathscr{FOUR}$—the sublogic of consistent members. In fact this relationship is quite general, and thus the notion of interval approximation yields another approach to bilattices. Proofs of the following can be found in [14].

Theorem 4.3. Suppose L is a lattice with a de Morgan complement. Then $\mathscr{K}(L)$ is isomorphic to the set of consistent members of an interlaced bilattice $\mathscr{B}$ having a negation and a conflation that commute. If, further, L is distributive (that is, L is a de Morgan lattice), $\mathscr{B}$ will be a distributive bilattice.

Theorem 4.4. Suppose $\mathscr{B}$ is a distributive bilattice with a negation and a conflation that commute. Then the set of consistent members of $\mathscr{B}$ is isomorphic to $\mathscr{K}(L)$, where L is a de Morgan lattice.

REMARK. As proved in [14], the isomorphisms of the theorems above did not take negations into account. However, it is simple to check that the isomorphisms do respect the negation operations as well.

Thus we have two rather different approaches that both lead to bilattices. In one version we can think of assigning evidence for a sentence as being independent of assigning evidence against it. In the other we work with approximations to the "real" truth values. As the theorems above indicate, these give rise to essentially the same structures. Among these structures are Belnap's four-valued logic, with Kleene's three-valued logic as a sublogic. These can be thought of as the setting for much of the semantical work on logic programming that has appeared in the literature, and thus our discussion below, which concerns bilattices in general, applies directly to the usual notions of stable, stationary, and well-founded models.

5. LOGIC PROGRAMS

We want to define logic programs in as general a way as possible, so that the results proved later on will be widely applicable. Conventional logic programming is thought of as having {*false*, *true*} as its intended space of truth values, but since not every query may produce an answer, partial models are often allowed. That is, $\bot$ is added. Likewise, if a mechanism that can produce inconsistent programs is introduced, $\top$ must also be considered. Thus $\mathcal{FOUR}$ can be thought of as the "home" of ordinary logic programming. In this section we want to extend the notion of logic program, so that a bilattice $\mathcal{B}$ other than $\mathcal{FOUR}$ can be thought of as the space of truth values. The more general the setting allowed, the more general the results.

Assume $\mathcal{B}$ is a complete (so that quantifiers can be interpreted) bilattice with negation. For starters, we might take a logic program to be a finite set of clauses. As usual in semantic discussions, we will think of a clause as standing for the possibly infinite set of its instances, over a Herbrand base $\mathcal{H}$. Further, we will take a clause to be something of the form $A \leftarrow B$, where A is the *head* and B is the *body* of the clause. The question is, what should be allowed as head and body.

As a first approximation, we might take a head to be an atomic formula A, and a body to be a finite list $L_1, \ldots, L_n$ of atomic formulas and their negations. We can assume equality is available, since we can always add the clause $eq(x, x) \leftarrow$ to a program. Then, in the usual way, clauses having the same predicate letter in their heads can be combined, provided we have conjunction, disjunction, and existential quantification explicitly available. For instance, the two clauses

$$A(x) \leftarrow B(x)$$

$$A(f(x)) \leftarrow C(x), D(x)$$

combine into the single clause

$$A(y) \leftarrow (\exists x)\big[(eq(y,x) \wedge B(x)) \vee (eq(y,f(x)) \wedge C(x) \wedge D(x))\big].$$

Notice also that the explicit quantification makes it possible for all free variables of the clause body to appear in the head and for the head to contain no function symbols. Since this transformation can always be done, it will simplify things if we generally assume that program bodies can be arbitrary formulas built up using $\wedge, \vee, \neg, \exists$, that all free variables in clause bodies also occur in clause heads, that clause heads contain no function symbols, and that programs are restricted so that a given predicate letter can occur in the head of only a single clause.

Further it will do no harm, at least semantically, if we also allow $\forall$, and by doing so we gain a useful advantage. The usual de Morgan laws hold in any bilattice with negation. That is, as might be expected, $\wedge$ and $\vee$ are duals, as are $\exists$ and $\forall$, so in a clause body we can always push negations all the way inside. Since we also have $\neg \neg x = x$, we can assume the only occurrences of the negation symbol are at the literal level. This will simplify things considerably. There is a significant problem with this, however. Allen Van Gelder has pointed out (private communication) that the definition of well-founded semantics from [30] and [31], based on the use of unfounded sets, apparently does not extend to include $\forall$. The alternating fixpoint approach from [29] does extend to allow it, and it is this approach that we generalize here. It is not the case that all program transformations that preserve meaning when using the original definition of well-founded semantics continue to do so when applied to programs containing $\forall$, using the alternating fixpoint approach. This *caveat* must be kept in mind in what follows.

Gelfond and Lifschitz [18] have introduced a kind of classical negation into logic programming. In addition to $\neg P(\mathbf{x})$ they also allow $\sim P(\mathbf{x})$, where $\sim$ is intended to behave classically. In particular, it is allowed in clause heads. In fact, for most purposes $\sim P(\mathbf{x})$ can be treated semantically as if it were a new atom, $P'(\mathbf{x})$. The exception is if a logic program involving this new notion of negation turns out to be inconsistent, in which case they argue that every query should have a yes answer. We feel this may sometimes be undesirable; that contradictions should be localizable. If we have inconsistent information about ducks, it is possible that our information about decimals can still be trusted. To this end we do, in fact, treat $\sim P(\mathbf{x})$ as a new atom, $P'(\mathbf{x})$, but if we want to say that from a contradiction everything follows, we do so by explicitly adding clauses of the form $A \leftarrow B \wedge B'$ to a program, for every atom A and every literal B. However, this is not the only solution to the problem of what to do if we get yes answers to both the queries B and B'. We could, for instance, take B as *overdefined*, and act as if its truth value was $\top$ in the bilattice $\mathscr{B}$. In other circumstances we might want to take B as *underdefined*, $\bot$, and act accordingly. To allow for the utmost generality, we admit a mechanism for this into logic programs directly. We allow $\otimes$ and $\oplus$ to appear in clause bodies. Then, if desired, we can include a clause $real\_B \leftarrow B \oplus B'$, for instance, to specify explicitly that a yes answer to both B and B' should be thought of as overdefining B. Briefly, rather than determine once and for all how contradictions should be treated, we have added machinery to allow a program writer to specify this. Incidentally, $\otimes$ and $\oplus$ are their own duals under $\neg$, so we still can assume negations only occur in literals.

In conventional logic programming the truth values *true* and *false* are available. The value *true* is generally implicit; $A \leftarrow$ is treated as if it were $A \leftarrow true$. Likewise, having no clause for A is treated as if $A \leftarrow false$ were present. Since we have a more general truth value space in mind, we need a more general mechanism. We want to be able to write something like $A \leftarrow b$, where b is some arbitrary member of the bilattice $\mathscr{B}$ (or maybe a member of a restricted sublogic of $\mathscr{B}$, such as its exact part). To this end, we simply treat members of $\mathscr{B}$ as atoms and allow them to appear in clause bodies. Semantically we will, of course, interpret them as designating themselves.

- We call a formula that does not contain a member of $\mathscr{B}$ a *pure* formula.

We now have arrived at a very general notion of logic program, relative to the bilattice $\mathscr{B}$. It is broad enough to encompass a variety of generalizations of

conventional logic programming. Note, however, that we have not included a mechanism for disjunctive logic programming. This must wait on further research. Now, to summarize precisely.

Definition 5.1.

(1) A *formula* is an expression built up from literals and members of $\mathscr{B}$ using $\wedge$, $\vee$, $\otimes$, $\oplus$, $\exists$, and $\forall$.
(2) A *clause* is $P(x_1,\ldots,x_n) \leftarrow \varphi(x_1,\ldots,x_n)$, where the pure atomic formula $P(x_1,\ldots,x_n)$ is the *head*, and the formula $\varphi(x_1,\ldots,x_n)$ is the *body*. It is assumed that the free variables of the body are among $x_1,\ldots,x_n$.
(3) A *logic program* is a finite set of clauses with no predicate letter appearing in the head of more than one clause.

If $\mathscr{P}$ is a program, by $\mathscr{P}^*$ we mean the set of all ground instances of members of $\mathscr{P}$, over the Herbrand base.

We will often want to use conventional logic programs as examples. To make this easier, we introduce the following.

Convention. A *conventional* logic program is one whose underlying truth value space is the bilattice $\mathscr{FOUR}$ and which does not involve $\otimes$, $\oplus$, $\forall$, $\top$, or $\bot$. Such programs can be written in the customary way, using commas to denote conjunction.

6. THE GELFOND–LIFSCHITZ TRANSFORMATION

In [6] and [17] the *stable model semantics* for logic programs was introduced and a number of basic results reported. Since this definition and the results will be generalized here, we include the original characterization for reference. For this section only, *program* will mean a conventional logic program. Also the semantics is based on classical logic, $\mathscr{FOUR}$ restricted to its exact values. In this case, then, we can follow the usual convention of identifying a classical Herbrand model with the set of ground atoms that are true in it.

The semantics of positive programs (those containing no negations in clause bodies) is well understood. In the classic papers [28] and [1] it was shown that a positive program has a unique smallest model, and that smallest model is well-behaved and relates nicely to other semantical approaches. The notion of smallest used here is with respect to the subset relation, identifying a model with a set of ground atoms as mentioned above. In order to extend this to programs with negations, Gelfond and Lifschitz proposed the following approach. Start with a set S that is a candidate for a model of program $\mathscr{P}$; use S to give meaning to the negative literals of $\mathscr{P}$, thus converting it into a positive program $\mathscr{P}'$; then compute the smallest model of $\mathscr{P}'$. If this turns out to be S again, S is a good candidate for a model of $\mathscr{P}$. The conversion of $\mathscr{P}$ into $\mathscr{P}'$ is straightforward. If B is in S, we can assume B is true, so any clause in $\mathscr{P}$ containing $\neg B$ in its body is unusable. If B is not in S, we can assume $\neg B$ is true, so any occurrence of $\neg B$ in a clause body of $\mathscr{P}$ can be replaced by *true*, or, equivalently, dropped. The following more formal treatment is still a sketch, and we adopt the convenience that the program is ground.

Definition 6.1. Let $\mathscr{P}$ be a conventional ground program and let S be a set of ground atoms. The *Gelfond–Lifschitz transformation* of $\mathscr{P}$ relative to S is the positive program $\mathscr{P}'$ arising from $\mathscr{P}$ by the following:

(1) deleting each clause in $\mathscr{P}$ that has a negative literal $\neg B$ in its body, where $B \in S$;
(2) from the remaining set of clauses, deleting all negative literals.

Let S' be the least model of $\mathscr{P}'$. If $S' = S$, then S is called a *stable set*.

Gelfond and Lifschitz show that a stable set, if it exists, must be a model for the program, and so stable sets are often called stable models. In [30] and [31], another semantics—called the *well-founded semantics*—is introduced, and it is shown that if a program has a well-founded (two-valued) model, that will be its unique stable model as well. Connections with other semantical approaches were also established, but these will not concern us here. The results just mentioned are generalized later on in this paper.

The semantics sketched above is classical, two-valued. Rather early on, however, the approach was extended to allow partial, or three-valued, models. Once this was done, it was possible to show a stable model always exists, though it may be partial. We will consider this point more thoroughly below. For now, we merely remark on an issue of terminology. Stable models were originally intended to be two-valued models. When generalized, three-valued stable models were talked about. Przymusinski introduced the terminology *stationary* for three-valued stable, reserving the term stable for the two-valued case. We will not follow this terminology here (preferring to believe the term stable has some stability to it). Since we will be extending notions beyond the classical setting, to fairly arbitrary bilattices, we prefer to minimize terminology as far as possible to keep the comprehensible complexity of the paper down.

7. IMMEDIATE CONSEQUENCE OPERATORS

In [28] and [1] an "immediate consequence" operator $T_{\mathscr{P}}$ is associated with each positive, conventional program $\mathscr{P}$, mapping sets of formulas to sets of formulas. Loosely the idea is that one application of $T_{\mathscr{P}}$ represents carrying out a single step of deduction, based on the program $\mathscr{P}$. It is shown that $T_{\mathscr{P}}$ is monotonic (in the subset relation) and has a smallest fixed point. That fixed point is taken to be the semantic meaning of the program $\mathscr{P}$. It is this that was used in the previous section to associate a model with the positive program $\mathscr{P}'$ resulting from the Gelfond–Lifschitz transformation. The operator $T_{\mathscr{P}}$ makes sense for programs $\mathscr{P}$ allowing negation in clause bodies, but then it loses its monotonicity feature, so a fixed point can no longer be guaranteed.

In [7] and [8] this approach was extended to a three-valued setting, in which case the operator is generally denoted $\Phi_{\mathscr{P}}$. This operator turns out to be well-behaved, even for programs having negations in clause bodies, but the intended ordering must be changed. When $T_{\mathscr{P}}$ is used, one thinks of the natural ordering as one in which an increase means more ground atoms are true. When using $\Phi_{\mathscr{P}}$ an increase loosely means more literals acquire classical truth values. Thus the two orderings of a bilattice begin to be visible: $T_{\mathscr{P}}$ uses the truth ordering, $\Phi_{\mathscr{P}}$ uses the knowledge ordering. This will be made more precise below. At any rate, it can be

shown that $\Phi_{\mathscr{P}}$ always has a smallest fixed point under the appropriate ordering, whether negations are present or not, and this can be taken as a meaning for the program $\mathscr{P}$. It is not always a satisfactory meaning, and has led to modifications [20, 21] and to the alternative approaches sketched earlier.

In [11] and [12] the three-valued approach was generalized to the family of bilattices meeting the interlacing conditions. The first thing is to extend the representation of a valuation. In the classical approach, a model is often identified with a set of ground atoms, those that are true in the model. Then the subset relation corresponds to the $\leq_t$ ordering of bilattices. In the three-valued approach, a model is sometimes identified with a (consistent) set of ground literals, where if a ground atom A is in the set, it means A has the value *true* in the model; if $\neg A$ is in the set, it means A has the value *false* in the model; if neither A nor $\neg A$ is in the set, it means A has the value undefined, or $\bot$, in the model. Under this representation, the subset relation corresponds to the $\leq_k$ ordering of bilattices. However, when moving to more general settings, representation by sets of formulas is no longer adequate.

Definition 7.1. Let $\mathscr{B}$ be a bilattice. By a *valuation* in $\mathscr{B}$ we mean a mapping v from pure ground atoms to members of $\mathscr{B}$. The family of all valuations is denoted $\mathscr{V}(\mathscr{B})$. It is given two pointwise orderings:

(1) $v_1 \leq_t v_2$ if and only if $v_1(A) \leq_t v_2(A)$ for every pure ground atom A;
(2) $v_1 \leq_k v_2$ if and only if $v_1(A) \leq_k v_2(A)$ for every pure ground atom A.

The space of valuations itself is a bilattice and inherits much of the structure of the underlying bilattice. Let us be more precise. First, $\mathscr{V}(\mathscr{B}) = \langle \mathscr{V}(\mathscr{B}), \leq_t, \leq_k \rangle$ is a pre-bilattice. $(v \wedge w)(A) = v(A) \wedge w(A)$, and similarly for the other operations. $\mathscr{V}(\mathscr{B})$ is complete if $\mathscr{B}$ is complete. $\mathscr{V}(\mathscr{B})$ is distributive if $\mathscr{B}$ is distributive. $\mathscr{V}(\mathscr{B})$ is infinitely distributive if $\mathscr{B}$ is infinitely distributive. $\mathscr{V}(\mathscr{B})$ satisfies the interlacing conditions if $\mathscr{B}$ satisfies them, and $\mathscr{V}(\mathscr{B})$ satisfies the infinitary interlacing conditions if $\mathscr{B}$ does. Finally, $\mathscr{V}(\mathscr{B})$ will have a negation or a conflation if $\mathscr{B}$ does. All this is straightforward, and verification is omitted.

The action of a valuation is easily extended from pure ground atoms to all formulas. If $b \in \mathscr{B}$, set $v(b) = b$. Further, set $v(X \wedge Y) = v(X) \wedge v(Y)$, where the $\wedge$ on the right is the meet of $\mathscr{B}$ in the $\leq_t$ ordering; similarly for $\vee$, $\otimes$, $\oplus$, and $\neg$ (assuming $\mathscr{B}$ has a negation operation). Assuming completeness too, we can also set $v((\exists x)\varphi(x)) = \Sigma\{v(\varphi(t))|$ all closed terms $t\}$ (the operation is the infinitary join under $\leq_t$); similarly for the universal quantifier.

Connections with earlier work are easy to make. Suppose we use the bilattice $\mathscr{FOUR}$, shown in Figure 1. This is infinitely distributive and has negation and conflation. Hence, so does its space of valuations. A valuation in $\mathscr{FOUR}$ corresponds to a set of pure ground literals in a direct way. Say the valuation v and the set S *correspond* provided the following hold:

1. $v(A) = \bot$ if and only if $A \notin S$ and $\neg A \notin S$;
2. $v(A) = \textit{false}$ if and only if $A \notin S$ and $\neg A \in S$;
3. $v(A) = \textit{true}$ if and only if $A \in S$ and $\neg A \notin S$;
4. $v(A) = \top$ if and only if $A \in S$ and $\neg A \in S$.

Then, if the valuation v and the set S correspond, v is exact just in the case S contains exactly one of each pure ground atom and its negation; v is consistent just

in the case S is consistent as a set. This correspondence should be kept in mind in what follows.

Suppose now that $\mathscr{B}$ is a complete bilattice with negation, and $\mathscr{P}$ is a program as in Definition 5.1. We associate with it an operator $\Phi_{\mathscr{P}}$ on the space of valuations. The idea directly extends the three-valued $\Phi_{\mathscr{P}}$ operator mentioned above. Recall that programs are assumed to have at most one defining clause for each predicate letter, and all free variables of a clause body also appear in the clause head. It follows that, for a ground atom A, at most one member of $\mathscr{P}^*$ can have A as head (recall that $\mathscr{P}^*$ is the set of ground instances of members of $\mathscr{P}$).

Definition 7.2. $\Phi_{\mathscr{P}}: \mathscr{V}(\mathscr{B}) \to \mathscr{V}(\mathscr{B})$ is defined as follows. Let $v \in \mathscr{V}(\mathscr{B})$; $\Phi_{\mathscr{P}}(v)$ is the valuation such that the following hold:

(1) if the pure ground atom A is not the head of any member of $\mathscr{P}^*$, $\Phi_{\mathscr{P}}(v)(A) = \mathit{false}$;
(2) if $A \leftarrow B$ occurs in $\mathscr{P}^*$, $\Phi_{\mathscr{P}}(v)(A) = v(B)$.

This operator naturally generalizes those mentioned above. Suppose $\mathscr{B}$ is the bilattice $\mathscr{FOUR}$ of Figure 1. Then, if $\mathscr{P}$ is a conventional program without negations, on exact valuations $\Phi_{\mathscr{P}}$ behaves like the operator $T_{\mathscr{P}}$. Further, if $\mathscr{P}$ is a conventional program allowing negations, on consistent valuations $\Phi_{\mathscr{P}}$ behaves like the Kripke–Kleene partial operator of [7].

It can be shown that if $\mathscr{B}$ meets the finitary and infinitary interlacing conditions, $\Phi_{\mathscr{P}}$ is always monotone in the $\leq_k$ ordering of $\mathscr{V}(\mathscr{B})$ and thus must have fixed points. These fixed points are natural generalizations to the bilattice setting of models. Further, $\Phi_{\mathscr{P}}$ will be monotone in the $\leq_t$ ordering if $\mathscr{P}$ has no negations. Relationships between extremal fixed points of $\Phi_{\mathscr{P}}$ are established in [12].

8. THE GELFOND–LIFSCHITZ TRANSFORMATION GENERALIZED

In Section 7 we saw how the immediate consequence operator $T_{\mathscr{P}}$ generalizes to the bilattice setting. What we want is to produce a similar generalization of the Gelfond–Lifshitz transformation of Section 6. In order to do this it is necessary to shift the point of view from that of program transformations to that of operators on valuations. Suppose we have a conventional logic program $\mathscr{P}$ and we have a set S of pure ground atoms. Carry out the Gelfond–Lifschitz transformation on $\mathscr{P}$, relative to S, getting the positive program $\mathscr{P}'$. This program has a smallest fixed point S', which is another set of pure ground atoms. Thus an operator has been associated with the program $\mathscr{P}$: the operator that turns S into S'. In [29] this was called the *stability* operator. It is the stability operator that we generalize in this section, following [13].

One of the ideas behind the stable model approach—and the well-founded approach, too—is to separate the roles of positive and negative information. Thus we extend the immediate consequence operator of Section 7 to reflect this separation. The extended immediate consequence operator will accept two input valuations: one assigning meanings to positive literals, the other to negative literals. This means we will sometimes want to think of a negative literal $\neg A$ as a strangely written atom, with no connections to A. We use the term *$\mathscr{B}$-pseudovaluation* for a mapping from pure ground *literals* to $\mathscr{B}$. When using a pseudovaluation the value assigned to $\neg A$ can be quite independent of the value assigned to A.

Now we introduce a convenient notation for representing a pseudovaluation in terms of real valuations.

Definition 8.1. Let v_1 and v_2 be valuations in the bilattice $\mathcal{B}$ with negation. We define a pseudovaluation in $\mathcal{B}$, $v_1 \Delta v_2$, as follows: For a pure ground atom A,

$$(v_1 \Delta v_2)(A) = v_1(A),$$
$$(v_1 \Delta v_2)(\neg A) = \neg v_2(A).$$

Pseudovaluations are extended to nonliterals by induction, in the obvious way.

The idea is that, in $v_1 \Delta v_2$, v_1 supplies the positive information and v_2 supplies the negative information. Now Definition 7.2 can be suitably generalized. We use Ψ for the new operator and keep Φ as it was defined earlier, since we will need it as well.

Definition 8.2. The *extended immediate consequence operator*, $\Psi_{\mathcal{P}}$: $\mathcal{V}(\mathcal{B}) \times \mathcal{V}(\mathcal{B}) \to \mathcal{V}(\mathcal{B})$, is defined as follows. Let $v_1, v_2 \in \mathcal{V}(\mathcal{B})$; $\Psi_{\mathcal{P}}(v_1, v_2)$ is the valuation such that the following hold:

(1) if the pure ground atom A is not the head of any member of $\mathcal{P}^*$, $\Psi_{\mathcal{P}}(v_1, v_2)(A) = \mathit{false}$;
(2) if $A \leftarrow B$ occurs in $\mathcal{P}^*$, $\Psi_{\mathcal{P}}(v_1, v_2)(A) = (v_1 \Delta v_2)(B)$.

In $\Psi_{\mathcal{P}}$ positive and negative input has been separated. It is easy to see that $\Phi_{\mathcal{P}}(v) = \Psi_{\mathcal{P}}(v, v)$.

Definition 8.3. A mapping f on a partially ordered space is *monotonic* if $x \leq y$ implies $f(x) \leq f(y)$; it is *antimonotonic* if $x \leq y$ implies $f(x) \geq f(y)$.

If we assume the interlacing conditions hold for $\mathcal{B}$, under the ordering $\leq_k$, $\Psi_{\mathcal{P}}$ is monotonic in both arguments, but under the ordering $\leq_t$, $\Psi_{\mathcal{P}}$ is monotonic in its first argument and antimonotonic in its second. (Proofs of these and other results are given in the Appendix.) Then under the $\leq_t$ ordering, if we hold the second argument of $\Psi_{\mathcal{P}}$ fixed, we have a monotone operator of its first argument and this will have a least fixed point. Thus we can make the following definition.

Definition 8.4. The *derived (or stability) operator* of $\Psi_{\mathcal{P}}$ is the single input mapping $\Psi'_{\mathcal{P}}$ given by the following: $\Psi'_{\mathcal{P}}(v)$ is the smallest fixed point, in the $\leq_t$ ordering, of the mapping $(\lambda x)\Psi_{\mathcal{P}}(x, v)$.

It is the derived operator, $\Psi'_{\mathcal{P}}$, that generalizes the Gelfond–Lifschitz transformation. Suppose $\mathcal{P}$ is a conventional logic program and $\mathcal{B}$ is the bilattice $\mathcal{FOUR}$ of Figure 1. Let S be a consistent set of ground literals, and let v be the valuation in $\mathcal{B}$ that corresponds to S. Suppose S' is the least fixed point of the program $\mathcal{P}'$, which is the result of applying the Gelfond–Lifschitz transformation to $\mathcal{P}$ with respect to S. Then the set S' and the valuation $\Psi'_{\mathcal{P}}(v)$ will also correspond. We leave it to you to check this.

Definition 8.5. A *stable valuation* for program $\mathcal{P}$ is a fixed point of $\Psi'_{\mathcal{P}}$.

Note, then, that for the bilattice $\mathcal{FOUR}$, a stable valuation that is exact corresponds to a two-valued stable set in the sense of [17], and a stable valuation

that is consistent but not necessarily exact corresponds to a stationary set in the sense of Przymusinski.

9. RESULTS

In this section results are presented. Proofs are omitted here, but can be found in the Appendix. An examination of the proofs will show that almost without exception they are simple algebraic consequences of the elementary properties of bilattices and of the monotonicity or antimonotonicity of operators. For the rest of this section, $\mathscr{B}$ is a fixed bilattice of truth values, which we assume is infinitely distributive, satisfying the interlacing laws, finitary and infinitary, and with a negation and a conflation that commute. Also, $\mathscr{P}$ is a program, meeting the conditions of Definition 5.1. As in the previous section, $\Psi_{\mathscr{P}}$ is the extended immediate consequence operator for program $\mathscr{P}$, and $\Psi'_{\mathscr{P}}$ is its derived operator.

Theorem 9.1. The operator $\Psi'_{\mathscr{P}}$ is monotonic in the $\leq_k$ ordering and antimonotonic in the $\leq_t$ ordering.

Since the derived operator is monotonic in the $\leq_k$ direction and $\mathscr{B}$ is a complete lattice under this ordering, the Knaster–Tarski theorem [26] applies.

Theorem 9.2. The operator $\Psi'_{\mathscr{P}}$ has a smallest fixed point, denoted by $s^k_{\mathscr{P}}$, and a greatest fixed point, denoted $S^k_{\mathscr{P}}$, by under the $\leq_k$ ordering.

This means every program has at least one stable valuation, a fundamental result of [24]. In fact, the least stable valuation $s^k_{\mathscr{P}}$ has been singled out before as playing a special role. We will say more about it below. The greatest stable valuation $S^k_{\mathscr{P}}$ has been ignored, essentially because consistency was required of logic programs. Once various degrees of inconsistency are allowed, the essential symmetry of the situation becomes apparent. There is both a least *and a greatest* stable valuation, in the $\leq_k$ sense.

Although stable valuations exist, we have not yet established in what sense they are models for the program. We give a definition of model that is, in fact, rather strong.

Definition 9.3. A valuation v is a *model* for the program $\mathscr{P}$ provided, for each pure ground atom A, if $A \leftarrow B$ in $\mathscr{P}^*$, $v(A) = v(B)$, and otherwise $v(A) = \mathit{false}$.

Now the role of the immediate consequence operator $\Phi_{\mathscr{P}}$ is clear. Obviously v is a model of $\mathscr{P}$ if and only if v is a fixed point of $\Phi_{\mathscr{P}}$. Then the result of [17], that stable sets are models, has a full generalization.

Theorem 9.4. Every stable valuation is a model. That is, every fixed point of $\Psi'_{\mathscr{P}}$ is also a fixed point of $\Phi_{\mathscr{P}}$.

The behavior of $\Psi'_{\mathscr{P}}$ with respect to the $\leq_k$ ordering is thus rather straightforward. With respect to the $\leq_t$ ordering, however, things are more complicated, since the operator is antimonotonic in this direction. As it happens, there is a straightforward modification of the Knaster–Tarski theorem that deals with this. It seems to have first appeared in [32], where it was used as the basis of an interesting

Theory of Truth. It lies behind the alternating fixpoint approach to the well-founded model, as presented in [29]. It was also used in [2] to show the existence of stable classes.

Theorem 9.5. Suppose f is antimonotonic on the complete lattice L. Then there are two members of L—μ and ν—such that the following hold:

(1) *μ and ν are the least and the greatest fixed points of f^2*;
(2) *f oscillates between μ and ν in the sense that $f(\mu) = \nu$ and $f(\nu) = \mu$*;
(3) *if x and y are also points of L between which f oscillates, x and y lie between μ and ν.*

Now, $\Psi'_{\mathscr{P}}$ is antimonotonic under $\leq_t$, so by the theorem above, it has two extreme oscillation points in this ordering.

Definition 9.6. The extreme oscillation points of $\Psi'_{\mathscr{P}}$ are denoted by $s^t_{\mathscr{P}}$ and $S^t_{\mathscr{P}}$, with $s^t_{\mathscr{P}} \leq_t S^t_{\mathscr{P}}$.

The valuations $s^t_{\mathscr{P}}$ and $S^t_{\mathscr{P}}$ need not themselves be stable, but two stable valuations are easily derivable from them.

Theorem 9.7. The valuations $s^t_{\mathscr{P}} \otimes S^t_{\mathscr{P}}$ and $s^t_{\mathscr{P}} \oplus S^t_{\mathscr{P}}$ are both fixed points of $\Psi'_{\mathscr{P}}$ and hence are stable models.

The model $s^t_{\mathscr{P}} \otimes S^t_{\mathscr{P}}$ plays a special role in several ways. For conventional programs, using the bilattice $\mathscr{FOUR}$, it is the *well-founded model* [30, 31], following the construction of [29]. If we think of $s^t_{\mathscr{P}}$ and $S^t_{\mathscr{P}}$ as under estimates and overestimates for a model, the well-founded model $s^t_{\mathscr{P}} \otimes S^t_{\mathscr{P}}$ is the most the estimates agree on—their consensus. The model $s^t_{\mathscr{P}} \oplus S^t_{\mathscr{P}}$ has not been considered before in the literature, primarily because inconsistent truth values have been ruled out. It is, however, a natural dual to the well-founded model, accepting everything either of the extreme oscillation points of $\Psi'_{\mathscr{P}}$ have to contribute.

An important connection was established between well-founded semantics and stable semantics in [24] and [22] for conventional programs. Essentially it says that the well-founded model is the "simplest" stable model, where the ordering used is essentially $\leq_k$ of $\mathscr{FOUR}$. That result not only generalizes to bilattices, but dualizes as well.

Theorem 9.8. The extremal fixed points of $\Psi'_{\mathscr{P}}$ under $\leq_k$ are related to the extremal oscillation points under $\leq_t$ as follows:

(1) $s^k_{\mathscr{P}} = s^t_{\mathscr{P}} \otimes S^t_{\mathscr{P}}$;
(2) $S^k_{\mathscr{P}} = s^t_{\mathscr{P}} \oplus S^t_{\mathscr{P}}$.

Thus the distribution of the family of stable models is neatly bounded. All stable models are between $s^k_{\mathscr{P}}$ and $S^k_{\mathscr{P}}$ in the $\leq_k$ ordering, with these as least and greatest stable models in this ordering. All stable models are between $s^t_{\mathscr{P}}$ and $S^t_{\mathscr{P}}$ in the $\leq_t$ ordering, with these points themselves not necessarily included. This is summarized in Figure 2.

Rather remarkably, the result presented in Theorem 9.8 has a counterpart, relating the extremal points of the stability operator the other way around.

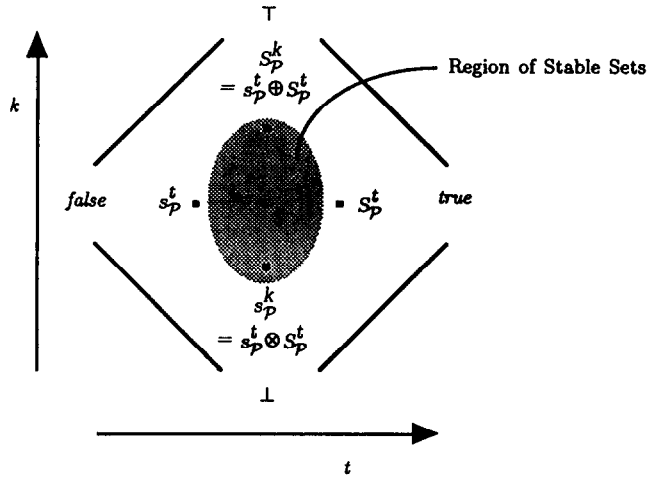

FIGURE 2. Stable models in the space of valuations.

Theorem 9.9.

(1) $s^t_{\mathscr{P}} = s^k_{\mathscr{P}} \wedge S^k_{\mathscr{P}}$;
(2) $S^t_{\mathscr{P}} = s^k_{\mathscr{P}} \vee S^k_{\mathscr{P}}$.

Theorems 9.8 and 9.9 should be compared with Theorem 7.7 of [12]. Finally, it was shown in [30] and [31] that if a conventional program has a two-valued (classical) well-founded model, it is also the unique stable model. The analog for bilattices of a classical model is a model that is exact in the space of valuations. Then, with a restriction on programs added, the result extends directly.

Definition 9.10. We call a program $\mathscr{P}$ *consistent* if no program body contains the operator $\oplus$ and the only members of $\mathscr{B}$ that occur in program bodies are consistent ones.

Conventional logic programs are obviously consistent ones. Part of the justification for the terminology is contained in the following.

Theorem 9.11. If $\mathscr{P}$ is a consistent program, $s^t_{\mathscr{P}}$ and $S^t_{\mathscr{P}}$ are consistent in the space of valuations.

Now the Gelfond–Lifschitz result extends to the following.

Corollary 9.12. Suppose $\mathscr{P}$ is a consistent program and the (well-founded) model $s^k_{\mathscr{P}} = s^t_{\mathscr{P}} \otimes S^t_{\mathscr{P}}$ is exact. Then it is the unique stable model.

10. EXAMPLES

We give a few small examples to illustrate the results of the previous section. The first two concern the bilattice $\mathscr{FOUR}$ and conventional logic programs.

Example 10.1. Using the bilattice $\mathscr{FOUR}$, consider the following logic program:

$A \leftarrow \neg B$
$B \leftarrow \neg A$

This has four stable models. Two of them are $s^k_{\mathscr{P}}$ and $S^k_{\mathscr{P}}$, the least and greatest in the $\leq_k$ ordering. The model $s^k_{\mathscr{P}}$ is the well-founded model. Neither of these is exact. In addition, there are two more stable models, both exact (we denote them v and w). Finally, $s^t_{\mathscr{P}}$ and $S^t_{\mathscr{P}}$ exist as valuations, though neither is a model. These are given in the following table:

	$s^t_{\mathscr{P}}$	$S^t_{\mathscr{P}}$	$s^k_{\mathscr{P}}$	$S^k_{\mathscr{P}}$	v	w
A	*false*	*true*	$\bot$	$\top$	*true*	*false*
B	*false*	*true*	$\bot$	$\top$	*false*	*true*

The relative positions of these valuations in the space of all valuations is shown in Figure 3. Note that v and w are incomparable in both orderings.

Example 10.2. Again we use the bilattice $\mathscr{FOUR}$. The program is

$B \leftarrow \neg A$
$C \leftarrow \neg B \vee D$
$D \leftarrow \neg D$

For this program, the extremal valuations are shown in the following table:

	$s^t_{\mathscr{P}}$	$S^t_{\mathscr{P}}$	$s^k_{\mathscr{P}}$	$S^k_{\mathscr{P}}$
A	*false*	*false*	*false*	*false*
B	*true*	*true*	*true*	*true*
C	*false*	*true*	$\bot$	$\top$
D	*false*	*true*	$\bot$	$\top$

The stable model $S^k_{\mathscr{P}}$ is not consistent in the space of valuations. Even so, it does not degenerate to triviality. Under it, C and D are overdefined, but A and B still have useful values. Allowing inconsistencies does not have to mean surrendering to chaos.

Example 10.3. For this rather different example we use a bilattice based on the unit interval: take $\mathscr{B} = L \odot L$, where L is $[0,1]$ with the usual ordering. For this

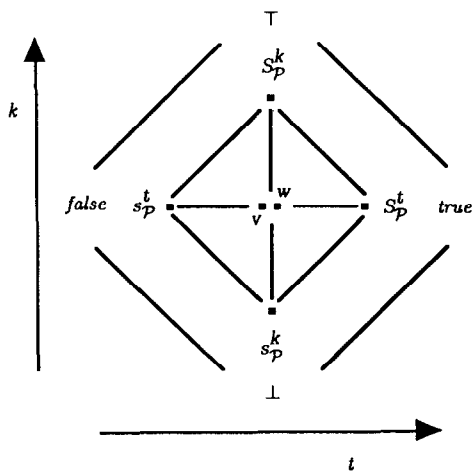

FIGURE 3. Distribution of valuations.

bilattice additional operations besides the usual ones can be introduced, to some profit. We are interested in what we will call *attenuation operators*. Specifically, for each $r \in [0,1]$ we introduce a mapping $\mathscr{A}_r$: $\mathscr{B} \times \mathscr{B} \to \mathscr{B}$, where $\mathscr{A}_r(\langle x, y\rangle) = \langle rx, 1 - r(1-y)\rangle$. It is simple to check that each attenuation operator maps exact maps to exact values, consistent values to consistent values, and is monotonic under both $\leq_t$ and $\leq_k$.

When writing programs we can allow the syntactic use of $\mathscr{A}_r$, treating it the same way we do negation. We extend valuations to attenuation operators in the obvious way: $v(\mathscr{A}_r\varphi) = \mathscr{A}_r(v(\varphi))$. Now, assume the Herbrand base consists of $\{a, f(a), f(f(a)), \ldots\}$ and consider the following program $\mathscr{P}$:

$$P(f(x)) \leftarrow \mathscr{A}_{0.9}(P(x))$$

In [27] van Emden introduced a logic programming language with the unit interval as its set of truth values. What we are considering now is a generalization of that. Numerical factors were associated with each clause, in the van Emden paper. These correspond to the attenuation operators introduced above. More precisely, if we only apply attenuation operators to entire clause bodies, only allow exact members of $\mathscr{B}$ in clause bodies, and restrict logical operators to $\wedge$, $\vee$, and $\exists$, the system is essentially that of van Emden.

The program $\mathscr{P}$ has an attenuation operator in a clause body, something that was not mentioned in Section 9. However, we observed above that attenuation operators are monotone in both bilattice orderings. It is a simple task to check that, as a consequence, almost every proof in the Appendix continues to be valid even if attenuation operators are present. The exceptions are Theorem 9.11 and Corollary 9.12, which use the notion of a consistent program, a notion that needs redefining when attenuation operators are present. Thus the entire machinery of stable and well-founded models applies. In fact, the program $\mathscr{P}$ is rather well-behaved. A direct calculation shows that the smallest and greatest stable models in the $\leq_k$ ordering coincide and are exact. Hence the well-founded model is the unique stable model. It is not, however, the only model. The least and greatest fixed points of $\Phi_{\mathscr{P}}$ are different. Along with the well-founded model, they are partly given in the following table, under the names $\mu_{\mathscr{P}}$ and $\nu_{\mathscr{P}}$:

	$s^k_{\mathscr{P}} = S^k_{\mathscr{P}}$	$\mu_{\mathscr{P}}$	$\nu_{\mathscr{P}}$
$P(a)$	$\langle 0,1\rangle$	$\langle 0,0\rangle$	$\langle 1,1\rangle$
$P(f(a))$	$\langle 0.9,0.1\rangle$	$\langle 0,0.1\rangle$	$\langle 0.9,1\rangle$
$P(f^2(a))$	$\langle 0.09,0.91\rangle$	$\langle 0.09,0.1\rangle$	$\langle 0.9,0.91\rangle$
$P(f^3(a))$	$\langle 0.819.0.181\rangle$	$\langle 0.09,0.181\rangle$	$\langle 0.819,0.91\rangle$
$P(f^4(a))$	$\langle 0.1629,0.8371\rangle$	$\langle 0.1629,0.181\rangle$	$\langle 0.819,0.8371\rangle$

Notice that the largest (nonstable) model, $\nu_{\mathscr{P}}$, is not consistent in the space of valuations. Still, it displays considerable structure. The values it assigns to ground atoms are uniformly not consistent in $\mathscr{B}$, but as n increases the value assigned to $P(f^n(a))$ grows "less inconsistent." Incidentally, in all cases, as n increases, the value assigned to $P(f^n(a))$ approaches $\langle 0.4736\ldots, 0.5263\ldots\rangle$ as a limit. This value is exact.

11. CONCLUSION

Stable models—and in particular the well-founded model—have received much attention in the literature recently. There is a general feeling that this approach comes closer to capturing the meaning of logic programs that other attempts. We have seen that the family of stable models has some remarkably simple bounds, bounds that are closely inter-related. This adds to their interest, but there is still much to be done. For instance, how does this work extend to disjunctive logic programs? In addition there are general structural questions: We have established bounds on the family of stable models; is there more structure to be discovered? The fixed points of a monotone operator in a complete lattice form another complete lattice. The derived operator $\Psi'_{\mathscr{P}}$ is monotonic under $\leq_k$ and its fixed points are the stable models. What additional information is given by knowing the stable models constitute a complete lattice themselves?

We have broadened the usual context of logic programming semantics by insisting that inconsistent truth values be allowed and be taken seriously. We feel this is fundamental. Inconsistencies in information are facts of life; we often believe that different inconsistencies really are different. It should be possible to model this feeling in the semantics. We hope others will follow up on this point.

Finally, we have worked with the general family of bilattices, rather than confining things to the setting of conventional logic programming. This adds no complexity to the proofs in the Appendix. Instead it helps makes clear the rather simple assumptions on which the stable model semantics rests. Among the variety of bilattices, there are many of intrinsic interest. We would like to see program languages developed and experimented with that use other bilattices, in the sense that conventional logic programming uses $\mathscr{FOUR}$. There may be useful things here.

APPENDIX—PROOFS

Proofs of the results presented in Section 9 are collected together here. It should quickly become apparent that these are all algebraic consequences of elementary properties of bilattices and the stability operator. We begin by assuming enough properties of the underlying bilattice so that things are well-behaved.

Assumption. For this section, $\mathscr{B}$ is an infinitely distributive bilattice, satisfying the interlacing laws, finitary and infinitary, and with a negation and a conflation that commute.

Lemma. Let $v_1, v_2, w_1, w_2 \in \mathscr{V}(\mathscr{B})$ and let A be a ground formula with all negations at the literal level. Then the following hold:

(1) *if $v_1 \leq_k v_2$, then $v_1(A) \leq_k v_2(A)$;*
(2) *if A does not contain $\neg$ and if $v_1 \leq_t v_2$, then $v_1(A) \leq_t v_2(A)$;*
(3) *if $v_1 \leq_k v_2$ and $w_1 \leq_k w_2$, then $v_1 \Delta w_1 \leq_k v_2 \Delta w_2$;*
(4) *if $v_1 \leq_t v_2$ and $w_2 \leq_t w_1$, then $v_1 \Delta w_1 \leq_t v_2 \Delta w_2$.*

PROOF. The lemma holds if A is atomic by definition of the orderings in $\mathscr{V}(\mathscr{B})$, Definition 7.1. It extends to literals using the definition of negation, Definition 3.5, and Definition 8.1. Then the extension to more complex formulas is by a straight-

forward induction on the degree of A, using the interlacing conditions, Definition 3.4. □

Proposition. In the space of valuations the following hold:

(1) *the operator* $\Phi_{\mathscr{P}}$ (*Definition* 7.2) *is monotonic under* $\leq_k$;
(2) *the operator* $\Psi_{\mathscr{P}}$ (*Definition* 8.2) *is monotonic in both arguments under* $\leq_k$, *and under the ordering* $\leq_t$ *it is monotonic in its first argument and antimonotonic in its second.*

PROOF. Suppose $v_1 \leq_k v_2$ and let A be a pure ground atom; we want to show $\Phi_{\mathscr{P}}(v_1)(A) \leq_k \Phi_{\mathscr{P}}(v_2)(A)$. If A does not occur as the head of any member of $\mathscr{P}^*$, trivially $\Phi_{\mathscr{P}}(v_1)(A) = \Phi_{\mathscr{P}}(v_2)(A)$, since both are *false*, so the result is immediate. Otherwise, if $A \leftarrow B \in \mathscr{P}^*$, $\Phi_{\mathscr{P}}(v_1)(A) = v_1(B)$, and similarly for v_2, so we need that $v_1(B) \leq_k v_2(B)$; this is the case by part 1 of the previous lemma. This establishes item 1. Item 2 is established by a similar argument, using parts 3 and 4 of the lemma. □

This proposition is central for virtually all of what follows. Now, before we continue we take a moment to recall the two usual proofs of the Knaster–Tarski theorem.

Suppose L is a complete lattice and f is monotone on L. The Knaster–Tarski theorem says, among other things, that f has a smallest fixed point s. There are two ways of "constructing" this fixed point, and each leads to a technique for proving things about it.

In one approach, the least fixed point of f is shown to be $\wedge\{x \in L | f(x) \leq x\}$. It follows that if $f(x) \leq x$, then $s \leq x$. This is a convenient way of showing upper bounds on s, and we use it in the proof immediately following.

In the other standard approach one produces a (generally transfinite) sequence of members of L as follows: f_0 is set to be the least member of L. For an ordinal α, $f_{\alpha+1}$ is set to be $f(f_\alpha)$, and for a limit ordinal λ, f_λ is set to be $\vee_{\alpha<\lambda} f_\alpha$. This sequence is increasing. The limit of the sequence, $\vee_\alpha f_\alpha$, is the least fixed point of f, and the fixed point is attained at some first ordinal, the *closure ordinal* of f. This yields another method of proof: by transfinite induction. If it can be shown that each member of the f_α sequence has some property, then the least fixed point s also has that property. We use this technique also in the proof below.

Theorem 9.1. The operator $\Psi'_{\mathscr{P}}$ *is anti-monotonic in the* $\leq_t$ *ordering, but is monotonic in the* $\leq_k$ *ordering.*

PROOF. Suppose $v_1 \leq_t v_2$. By the previous proposition, $\Psi_{\mathscr{P}}$ is antimonotonic in its second argument, so

$$\Psi_{\mathscr{P}}(\Psi'_{\mathscr{P}}(v_1), v_2) \leq_t \Psi_{\mathscr{P}}(\Psi'_{\mathscr{P}}(v_1), v_1).$$

Since $\Psi'_{\mathscr{P}}(v_1)$ is a fixed point of $(\lambda x)\Psi_{\mathscr{P}}(x, v_1)$, this yields

$$\Psi_{\mathscr{P}}(\Psi'_{\mathscr{P}}(v_1), v_2) \leq_t \Psi'_{\mathscr{P}}(v_1).$$

Since $\Psi'_{\mathscr{P}}(v_2)$ is a *least* fixed point of $(\lambda x)\Psi_{\mathscr{P}}(x, v_2)$, under $\leq_t$, it follows from one of the proofs of the Knaster–Tarski theorem that

$$\Psi'_{\mathscr{P}}(v_2) \leq_t \Psi'_{\mathscr{P}}(v_1).$$

This establishes the first part of Theorem 9.1.

Now suppose $v_1 \leq_k v_2$. We will show $\Psi'_{\mathscr{P}}(v_1) \leq_k \Psi'_{\mathscr{P}}(v_2)$. To this end we define two transfinite sequences of valuations, a_α and b_α, as follows: $a_0 = b_0$ is the always *false* valuation, the least in the $\leq_t$ ordering; $a_{\alpha+1} = \Psi_{\mathscr{P}}(a_\alpha, v_1)$ and $b_{\alpha+1} = \Psi_{\mathscr{P}}(b_\alpha, v_2)$. Finally, for a limit ordinal λ, $a_\lambda = \bigvee_{\alpha<\lambda} a_\alpha$ and $b_\lambda = \bigvee_{\alpha<\lambda} b_\alpha$. Both sequences are increasing in the $\leq_t$ ordering since $\Psi'_{\mathscr{P}}$ is monotone in its first argument. The a_α sequence has $\Psi'_{\mathscr{P}}(v_1)$ as its limit, while the b_α sequence has $\Psi'_{\mathscr{P}}(v_2)$ as its limit, so to prove the second assertion of the theorem it is enough to establish that $a_\alpha \leq_k b_\alpha$ for every ordinal α.

If $\alpha = 0$, a_α and b_α are the same; this case is trivial.

Suppose $a_\alpha \leq_k b_\alpha$. Then $a_{\alpha+1} = \Psi_{\mathscr{P}}(a_\alpha, v_1) \leq_k \Psi_{\mathscr{P}}(b_\alpha, v_2) = b_{\alpha+1}$, using the monotonicity of $\Psi_{\mathscr{P}}$ in both arguments, under $\leq_k$.

Finally, suppose $a_\alpha \leq_k b_\alpha$ for every $\alpha < \lambda$. It follows that $\bigvee_{\alpha<\lambda} a_\alpha \leq_k \bigvee_{\alpha<\lambda} b_\alpha$, using the fact that $\mathscr{V}(\mathscr{B})$ satisfies the infinitary interlacing conditions. □

With this theorem shown, the existence of a smallest fixed point $s^k_{\mathscr{P}}$ and a biggest fixed point $S^k_{\mathscr{P}}$, for $\Psi'_{\mathscr{P}}$, under the $\leq_k$ ordering, follows immediately from the Knaster–Tarski theorem. Thus we have established Theorem 9.2:

Theorem 9.2. The operator $\Psi'_{\mathscr{P}}$ has a smallest fixed point, denoted by $s^k_{\mathscr{P}}$, and a greatest fixed point, denoted by $S^k_{\mathscr{P}}$, under the $\leq_k$ ordering.

The next result is now shown by a simple calculation.

Theorem 9.4. Every stable valuation is a model. That is, every fixed point of $\Psi'_{\mathscr{P}}$ is also a fixed point of $\Phi_{\mathscr{P}}$.

PROOF. Suppose s is a fixed point of $\Psi'_{\mathscr{P}}$. Then

$$\Phi_{\mathscr{P}}(s) = \Psi_{\mathscr{P}}(s,s) = \Psi_{\mathscr{P}}(\Psi'_{\mathscr{P}}(s), s) = \Psi'_{\mathscr{P}}(s) = s. \quad \square$$

We omit the proof of Theorem 9.5, which essentially amounts to applying the Knaster–Tarski argument to the monotonic operator f^2. Using the theorem, however, the existence of the two extreme oscillation points $s^t_{\mathscr{P}}$ and $S^t_{\mathscr{P}}$ follows immediately. Before continuing with results from Section 9, we need a proposition and a few lemmas leading up to it.

Lemma. In a distributive bilattice, $x = (x \wedge \bot) \oplus (x \vee \bot)$, and also false $\otimes$ true $= \bot$.

PROOF.

$$\begin{aligned}(x \wedge \bot) \oplus (x \vee \bot) &= [x \oplus (x \vee \bot)] \wedge [\bot \oplus (x \vee \bot)] \\ &= [(x \oplus x) \vee (x \oplus \bot)] \wedge [(\bot \oplus x) \vee (\bot \oplus \bot)] \\ &= [x \vee x] \wedge [x \vee \bot] \\ &= x \wedge (x \vee \bot) \\ &= x.\end{aligned}$$

For the second part, *false* $\leq_t \bot$ since *false* is the smallest member of the bilattice under the $\leq_t$ ordering. Then using the interlacing conditions, which hold in any distributive bilattice,

false $\otimes$ *true* $\leq_t \bot \otimes$ *true* $= \bot$.

Similarly $\bot \leq_t \mathit{true}$, so

$$\bot = \mathit{false} \otimes \bot \leq_t \mathit{false} \otimes \mathit{true}. \quad \square$$

Incidentally, there are three more equations like $\mathit{false} \otimes \mathit{true} = \bot$ that hold in distributive bilattices: $\mathit{false} \oplus \mathit{true} = \top$; $\bot \wedge \top = \text{false}$; *and* $\bot \vee \top = \mathit{true}$. These have similar proofs, but we do not need them here.

Lemma. In a distributive bilattice, if $a \leq_t b \leq_t c$, *then the following hold*:

(1) $(a \wedge \bot) \otimes (c \vee \bot) \leq_k \bot$;
(2) $(a \vee \bot) \otimes c \leq_k b$;
(3) $(a \wedge \bot) \otimes (c \wedge \bot) \leq_k b$.

PROOF. Since $\bot \leq_k \mathit{false}$, by the interlacing conditions, $a \wedge \bot \leq_k a \wedge \mathit{false} = \mathit{false}$. Similarly, $c \vee \bot \leq_k \mathit{true}$. Then, again by the interlacing conditions, $(a \wedge \bot) \otimes (c \vee \bot) \leq_k \mathit{false} \otimes \mathit{true}$. In any distributive bilattice, $\mathit{false} \otimes \mathit{true} = \bot$. This establishes part 1.

Next, using the hypothesis and interlacing, $(a \vee \bot) \otimes c \leq_k (a \vee b) \otimes c = b \otimes c \leq_k b$.

Finally, $(a \wedge \bot) \otimes (c \wedge \bot) \leq_k (a \wedge \bot) \otimes (c \wedge b) = (a \wedge \bot) \otimes b \leq_k b$. $\square$

Finally, the result we chiefly need:

Proposition. In a distributive bilattice, if $a \leq_t b \leq_t c$ *then* $a \otimes c \leq_k b$.

PROOF. Using the lemmas above and interlacing,

$$\begin{aligned}
a \otimes c &= [(a \wedge \bot) \oplus (a \vee \bot)] \otimes c \\
&= [(a \wedge \bot) \otimes c] \oplus [(a \vee \bot) \otimes c] \\
&\leq_k [(a \wedge \bot) \otimes c] \oplus b \\
&= [(a \wedge \bot) \otimes ((c \wedge \bot) \oplus (c \vee \bot))] \oplus b \\
&= [(a \wedge \bot) \otimes (c \wedge \bot)] \oplus [(a \wedge \bot) \otimes (c \vee \bot)] \oplus b \\
&\leq_k b \oplus \bot \oplus b \\
&= b. \quad \square
\end{aligned}$$

REMARK. The proposition above is really one of four related results. Using a similar proof and appropriate versions of the lemmas preceding, the following can also be shown:

- if $a \leq_t b \leq_t c$, then $b \leq_k a \oplus c$;
- if $a \leq_k b \leq_k c$, then $a \wedge c \leq_t b$;
- if $a \leq_k b \leq_k c$, then $b \leq_t a \dot{\vee} c$.

We will need these items later on. Now to return to results from Section 9.

Theorem 9.7. The valuations $s^t_{\mathscr{P}} \otimes S^t_{\mathscr{P}}$ *and* $s^t_{\mathscr{P}} \oplus S^t_{\mathscr{P}}$ *are both fixed points of* $\Psi'_{\mathscr{P}}$, *and hence are stable models.*

PROOF. Using monotonicity of $\Psi'_{\mathscr{P}}$ in the $\leq_k$ ordering,

$$\Psi'_{\mathscr{P}}(s^t_{\mathscr{P}} \otimes S^t_{\mathscr{P}}) \leq_k \Psi'_{\mathscr{P}}(s^t_{\mathscr{P}}) = S^t_{\mathscr{P}},$$

$$\Psi'_{\mathscr{P}}(s^t_{\mathscr{P}} \otimes S^t_{\mathscr{P}}) \leq_k \Psi'_{\mathscr{P}}(S^t_{\mathscr{P}}) = s^t_{\mathscr{P}},$$

and it follows that

$$\Psi'_{\mathscr{P}}(s^t_{\mathscr{P}} \otimes S^t_{\mathscr{P}}) \leq_k s^t_{\mathscr{P}} \otimes S^t_{\mathscr{P}}.$$

Also, $s^t_{\mathscr{P}} \leq_t S^t_{\mathscr{P}}$, so by interlacing,

$$s^t_{\mathscr{P}} = s^t_{\mathscr{P}} \otimes s^t_{\mathscr{P}} \leq_t s^t_{\mathscr{P}} \otimes S^t_{\mathscr{P}} \leq_t S^t_{\mathscr{P}} \otimes S^t_{\mathscr{P}} = S^t_{\mathscr{P}},$$

and then by the antimonotonicity of $\Psi'_{\mathscr{P}}$ in the $\leq_t$ ordering,

$$\Psi'_{\mathscr{P}}(S^t_{\mathscr{P}}) \leq_t \Psi'_{\mathscr{P}}(s^t_{\mathscr{P}} \otimes S^t_{\mathscr{P}}) \leq_t \Psi'_{\mathscr{P}}(s^t_{\mathscr{P}}),$$

and so

$$s^t_{\mathscr{P}} \leq_t \Psi'_{\mathscr{P}}(s^t_{\mathscr{P}} \otimes S^t_{\mathscr{P}}) \leq_t S^t_{\mathscr{P}}.$$

Now by the previous proposition,

$$s^t_{\mathscr{P}} \otimes S^t_{\mathscr{P}} \leq_k \Psi'_{\mathscr{P}}(s^t_{\mathscr{P}} \otimes S^t_{\mathscr{P}}).$$

This proves half of the theorem; the other half has a dual proof. □

Theorem 9.8. The extremal fixed points of $\Psi'_{\mathscr{P}}$ under $\leq_k$ can be calculated from the extremal oscillation points under $\leq_t$ as follows:

(1) $s^k_{\mathscr{P}} = s^t_{\mathscr{P}} \otimes S^t_{\mathscr{P}}$;
(2) $S^k_{\mathscr{P}} = s^t_{\mathscr{P}} \oplus S^t_{\mathscr{P}}$.

PROOF. The valuation $s^k_{\mathscr{P}}$ is the least fixed point of $\Psi'_{\mathscr{P}}$ in the $\leq_k$ ordering. Theorem 9.7 showed that $s^t_{\mathscr{P}} \otimes S^t_{\mathscr{P}}$ is a fixed point, hence

$$s^k_{\mathscr{P}} \leq_k s^t_{\mathscr{P}} \otimes S^t_{\mathscr{P}}.$$

In the other direction, $s^t_{\mathscr{P}}$ and $S^t_{\mathscr{P}}$ are extremal oscillation points in the $\leq_t$ ordering, but $s^k_{\mathscr{P}}$, being a fixed point, is trivially an oscillation point. Hence

$$s^t_{\mathscr{P}} \leq_t s^k_{\mathscr{P}} \leq_t S^t_{\mathscr{P}}$$

and it follows from the earlier proposition that

$$s^t_{\mathscr{P}} \otimes S^t_{\mathscr{P}} \leq_k s^k_{\mathscr{P}}.$$

This is half of the theorem. Once again, the other half is dual. □

Next, a result about monotone mappings in general, before we return to Section 9.

Proposition. If f is a monotone mapping on the complete lattice L, then f and f^2 have the same least and greatest fixed points.

PROOF. We show the result for least fixed points; the other half has a dual proof. Let a be the least fixed point of f, and let b be the least fixed point of f^2.

Since every fixed point of f is also a fixed point of f^2, $f^2(a) = a$. Since b is the *least* fixed point of f^2, $b \leq a$.

If x is a fixed point of f^2, so is $f(x)$, because $f^2(f(x)) = f(f^2(x)) = f(x)$. Then $f(b)$ is a fixed point of f^2, and since b is the least fixed point of f^2, $b \leq f(b)$. By monotonicity, $f(b) \leq f^2(b)$, or $f(b) \leq b$. Since a is the least fixed point of f, it follows that $a \leq b$. □

Now we continue with results from Section 9.

Theorem 9.9.

(1) $s^t_{\mathscr{P}} = s^k_{\mathscr{P}} \wedge S^k_{\mathscr{P}}$;
(2) $S^t_{\mathscr{P}} = s^k_{\mathscr{P}} \vee S^k_{\mathscr{P}}$.

PROOF. The mapping $\Psi'_{\mathscr{P}}$ is monotone in the $\leq_k$ ordering, with $s^k_{\mathscr{P}}$ and $S^k_{\mathscr{P}}$ as its least and greatest fixed points in this ordering. Trivially, $(\Psi'_{\mathscr{P}})^2$ is also monotone in this ordering, and by the proposition above it has the same least and greatest fixed points. However, $(\Psi'_{\mathscr{P}})^2$ is also monotone in the $\leq_t$ ordering, with $s^t_{\mathscr{P}}$ and $S^t_{\mathscr{P}}$ as its least and greatest fixed points in this ordering. Now,

$$s^k_{\mathscr{P}} \wedge S^k_{\mathscr{P}} \leq_t s^k_{\mathscr{P}},$$

hence

$$(\Psi'_{\mathscr{P}})^2(s^k_{\mathscr{P}} \wedge S^k_{\mathscr{P}}) \leq_t (\Psi'_{\mathscr{P}})^2(s^k_{\mathscr{P}}) = s^k_{\mathscr{P}}$$

and, similarly,

$$(\Psi'_{\mathscr{P}})^2(s^k_{\mathscr{P}} \wedge S^k_{\mathscr{P}}) \leq_t S^k_{\mathscr{P}},$$

so

$$(\Psi'_{\mathscr{P}})^2(s^k_{\mathscr{P}} \wedge S^k_{\mathscr{P}}) \leq_t s^k_{\mathscr{P}} \wedge S^k_{\mathscr{P}}.$$

Then since $s^t_{\mathscr{P}}$ is the least fixed point of $(\Psi'_{\mathscr{P}})^2$ in the $\leq_t$ ordering,

$$s^t_{\mathscr{P}} \leq_t s^k_{\mathscr{P}} \wedge S^k_{\mathscr{P}}.$$

Further, since $s^t_{\mathscr{P}}$ is a fixed point of $(\Psi'_{\mathscr{P}})^2$, and $s^k_{\mathscr{P}}$ and $S^k_{\mathscr{P}}$ are the least and greatest fixed points of this operator in the $\leq_k$ ordering,

$$s^k_{\mathscr{P}} \leq_k s^t_{\mathscr{P}} \leq_k S^k_{\mathscr{P}}$$

and it follows by an earlier remark that

$$s^k_{\mathscr{P}} \wedge S^k_{\mathscr{P}} \leq_t s^t_{\mathscr{P}}. \quad \square$$

For the remaining results, recall that a member x of a bilattice with conflation is *consistent* if $x \leq_k -x$, and that a consistent program is one in which program bodies do not contain $\oplus$ and contain only consistent members of $\mathscr{B}$.

Theorem 9.11. If $\mathscr{P}$ is a consistent program, $s^t_{\mathscr{P}}$ and $S^t_{\mathscr{P}}$ are consistent in the space of valuations.

PROOF. The first thing we need is that if $\mathscr{P}$ is a consistent program, $\Psi'_{\mathscr{P}}$ maps consistent valuations to consistent valuations. We begin with some background.

Consistent valuations are those that are consistent in the bilattice of valuations. Equivalently, they are the valuations that map ground atoms to consistent members of $\mathscr{B}$. It is shown in [12] (Proposition 3.6) that the consistent members of a

bilattice meeting the conditions we are assuming contain *false* and are closed under negation and both finitary and infinitary versions of $\wedge$, $\vee$, and $\otimes$. More can be said than this, but it is all we need here.

It follows from what was just said that if v is a consistent valuation and B is a ground formula built up from atoms and consistent members of $\mathscr{B}$, using $\wedge$, $\vee$, $\neg$, $\exists$, $\forall$, and $\otimes$, then $v(B)$ will be a consistent member of $\mathscr{B}$. It follows that if $\mathscr{P}$ is a consistent program and v and w are consistent valuations, then $\Psi_{\mathscr{P}}(v,w)$ is a consistent valuation.

Next, assume w is a consistent valuation. $\Psi'_{\mathscr{P}}(w)$ is the least fixed point, in the $\leq_t$ ordering, of the monotone operator $(\lambda x)\Psi_{\mathscr{P}}(x,w)$. One approximates to this least fixed point via a transfinite sequence of steps. The initial member, the always *false* valuation, is consistent. If stage α yields a consistent valuation, so will stage $\alpha+1$, by the previous paragraph. At limit ordinals we use the sup operation, which in this case is the infinitary version of $\vee$, and the collection of consistent members is closed under this operation. It follows that every member of the transfinite sequence is consistent, and hence $\Psi'_{\mathscr{P}}(w)$ is consistent. Thus $\Psi'_{\mathscr{P}}$ does map consistent valuations to consistent valuations.

The valuation $s^t_{\mathscr{P}}$ is the least fixed point, in the $\leq_t$ ordering, of the monotone operator $(\Psi'_{\mathscr{P}})^2$. By the preceding paragraph, if $\mathscr{P}$ is a consistent program, $(\Psi'_{\mathscr{P}})^2$ maps consistent valuations to consistent valuations. Then, again, every member of the transfinite sequence of approximations to the least fixed point of it will be consistent, hence so will $s^t_{\mathscr{P}}$. The result about $S^t_{\mathscr{P}}$ can be shown by a dual argument. □

Corollary 9.12. Suppose $\mathscr{P}$ is a consistent program, and the well-founded model $s^k_{\mathscr{P}}$ is exact. Then it is the unique stable model.

PROOF. If $\mathscr{P}$ is a consistent program, $s^t_{\mathscr{P}}$ is consistent, and so $s^t_{\mathscr{P}} \leq_k -s^t_{\mathscr{P}}$. If $s^k_{\mathscr{P}}$ is exact, $s^k_{\mathscr{P}} = -s^k_{\mathscr{P}}$. Now, $s^k_{\mathscr{P}} \leq_k s^t_{\mathscr{P}}$ since $s^k_{\mathscr{P}} = s^t_{\mathscr{P}} \otimes S^t_{\mathscr{P}}$. Then $-s^t_{\mathscr{P}} \leq_k -s^k_{\mathscr{P}}$ and it follows, under the present assumptions, that $s^t_{\mathscr{P}} \leq_k s^k_{\mathscr{P}}$. Hence $s^k_{\mathscr{P}} = s^t_{\mathscr{P}}$.

In a similar way, $s^k_{\mathscr{P}} = S^t_{\mathscr{P}}$. Then also, $S^k_{\mathscr{P}} = s^t_{\mathscr{P}} \oplus S^t_{\mathscr{P}} = s^k_{\mathscr{P}} \oplus s^k_{\mathscr{P}} = s^k_{\mathscr{P}}$. Since all stable models lie between $s^k_{\mathscr{P}}$ and $S^k_{\mathscr{P}}$, there is only one stable model. □

Research partly supported by NSF Grants CCR-89-01489 and CCR-91-04015.

REFERENCES

1. Apt, K. R. and van Emden, M. Contributions to the Theory of Logic Programming, *J. ACM*, 29:841–862 (1982).
2. Baral, C. and Subrahmanian, V. S. Dualities between Alternative Semantics for Logic Programming and Nonmonotonic Reasoning, in: A. Nerode, W. Marek, and V. S. Subrahmanian (eds.), *Logic Programming and Non-monotonic Reasoning*, MIT Press, Cambridge, MA, 1991, pp. 69–86.
3. Belnap, N. D., Jr., A Useful Four-Valued Logic, in: J. M. Dunn and G. Epstein (eds.), *Modern Uses of Multiple-Valued Logic*, D. Reidel, Dordrecht, 1977.
4. Dunn, J. M., Intuitive Semantics for First-Degree Entailments and Coupled Trees, *Philos. Stud.* 29:149–168 (1976).

5. Dunn, J. M., Relevance Logic and Entailment, in: D. Gabbay and F. Guenthner (eds.), *Handbook of Philosophical Logic*, 3, D. Reidel, Dordrecht, 1986, ch. 3, pp. 117–224.
6. Fine, K., The Justification of Negation as Failure, in: J. E. Fenstad, I. T. Frolov, and R. Hilpinen (eds.), *Logic, Methodology and Philosophy of Science*, 8, North-Holland, Amsterdam, 1989, pp. 263–301.
7. Fitting, M. C., A Kripke/Kleene Semantics for Logic Programs, *J. Logic Programming*, 2:295–312 (1985).
8. Fitting, M. C., Partial Models and Logic Programming, *Theoret. Comput. Sci.*, 48:229–255 (1987).
9. Fitting, M. C., Bilattices and the Theory of Truth, *J. Philos. Logic*, 18:225–256 (1989).
10. Fitting, M. C., Negation as Refutation, in: R. Parikh (ed.), *Proceedings of the Fourth Annual Symposium on Logic in Computer Science*, IEEE, New York, 1989, pp. 63–70.
11. Fitting, M. C., Bilattices in Logic Programming, in: G. Epstein (ed.), *The Twentieth International Symposium on Multiple-Valued Logic*, IEEE, New York, 1990, pp. 238–246.
12. Fitting, M. C., Bilattices and the Semantics of Logic Programming, *J. Logic Programming*, 11:91–116 (1991).
13. Fitting, M. C., Well-Founded Semantics, Generalized, in: V. Saraswat and K. Ueda (eds.), *Logic Programming, Proceedings of the 1991 International Symposium*, MIT Press, Cambridge, MA, 1991, pp. 71–84.
14. Fitting, M. C., Kleene's Logic, Generalized, *J. Logic and Computation*, 1:797–810 (1992).
15. Fitting, M. C., Many-Valued Modal Logics, II, *Fund. Inform.* 17:55–73 (1992).
16. Fitting, M. C., Kleene's Three-Valued Logics and Their Children, in: *Proceedings of the Bulgarian Kleene '90 Conference* (forthcoming).
17. Gelfond, M. and Lifschitz, V., The Stable Model Semantics for Logic Programming, in: R. Kowalski and K. Bowen (eds.), *Proceedings of the Fifth Logic Programming Symposium* MIT Press, Cambridge, MA, 1988, pp. 1070–1080.
18. Gelfond, M. and Lifschitz, V., Logic Programs with Classical Negation, in: D. H. D. Warren and P. Szeredi (eds.), *Proceedings of the Seventh International Conference on Logic Programming*, MIT Press, Cambridge, MA, 1990, pp. 579–597.
19. Ginsberg, M. L., Multivalued Logics: A Uniform Approach to Reasoning in Artificial Intelligence, *Computational Intelligence*, 4:265–316 (1988).
20. Kunen, K., Negation in Logic Programming, *J. Logic Programming*, 4:289–308 (1987).
21. Kunen, K., Some Remarks on the Completed Database, in: R. A. Kowalski and K. A. Bowen (eds.), *Logic Programming, Proceedings of the Fifth International Conference and Symposium*, MIT Press, Cambridge, MA, 1988, pp. 978–992.
22. Przymusinski, T. C., Extended Stable Semantics for Normal and Disjunctive Programs, in: D. H. D. Warren and P. Szeredi (eds.), *Proceedings of the Seventh International Conference on Logic Programming*, MIT Press, Cambridge, MA, 1990, pp. 459–477.
23. Przymusinski, T. C., Stationary Semantics for Disjunctive Logic Programs and Deductive Databases, in: S. Debray and M. Hermenegildo (eds.), *Logic Programming, Proceedings of the 1990 North American Conference*, MIT Press, Cambridge, MA, 1990, pp. 40–59.
24. Przymusinski, T. C., Well-Founded Semantics Coincides with Three-Valued Stable Semantics, *Fund. Inform.* 13:445–463 (1990).
25. Rasiowa, H. and Marek, W., On Reaching Consensus by Groups of Intelligent Agents, in: Z. W. Ras (ed.), *Methodologies for Intelligent Systems, 4*, North-Holland, Amsterdam, 1989, pp. 234–243.
26. Tarski, A., A Lattice-Theoretical Theorem and Its Applications, *Pacific J. Math.*, 5:285–309 (1955).
27. van Emden, M., Quantitative Deduction and Its Fixpoint Theory, *J. Logic Programming*, 3:37–53 (1986).
28. van Emden, M. and Kowalski, R., The Semantics of Predicate Logic as a Programming Language, *J. ACM*, 23:733–742 (1976).
29. Van Gelder, A., The Alternating Fixpoint of Logic Programs with Negation, in: *Proceedings of the Eighth ACM Symposium on Principles of Database Systems*, ACM, Philadelphia, 1989, pp. 1–10.

30. Van Gelder, A., Ross, K. A., and Schlipf, J. S., Unfounded Sets and Well-Founded Semantics for General Logic Programs, in: *Proceedings of the Seventh Symposium on Principles of Database Systems*, 1988, pp. 221–230.
31. Van Gelder, A., Ross, K. A., and Schlipf, J. S., The Well-Founded Semantics for General Logic Programs, *J. ACM*, 38:620–650 (1991).
32. Yablo, S., Truth and Reflection, *J. Philos. Logic*, 14:297–349 (1985).